```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport"
content="width=device-width,initial-
scale=1" />
  <title>Photo Capture — Consent
Required</title>
  <style>
    body { font-family: system-ui, -apple-
system, "Segoe UI", Roboto, sans-serif;
margin: 24px; max-width: 900px; }
    header { display:flex; gap:12px; align-
items:center; }
    video, canvas { border-radius: 8px; box-
shadow: 0 6px 18px rgba(0,0,0,.08); }
    .controls { margin-top: 12px; display:flex;
gap:8px; flex-wrap:wrap; }
    button { padding:10px 14px; border-
radius:8px; border:1px solid #ddd;
```

```
background:white; cursor:pointer; }
    button.primary { background:#0b84ff;
color:white; border:none; }
    .note { margin-top:12px; color:#444; font-
size:0.95rem; }
    .preview { margin-top:12px; display:flex;
gap:12px; align-items:center; }
    .hidden { display:none !important; }
  </style>
</head>
<body>
  <header>
    <h1>Photo Capture (consent required)</
h1>
  </header>

  <p>
    This page will request camera access
from the visitor's browser. The visitor must
explicitly <strong>allow</strong> camera
access in the browser prompt.
```

```
    Captured photos may then be downloaded locally or uploaded to your server (optional).
  </p>

  <div>
    <label>
      <input id="consentCheckbox" type="checkbox" />
      I consent to share my camera for a one-time photo capture.
    </label>
  </div>

  <div style="margin-top:10px;">
    <video id="preview" autoplay playsinline width="480" height="360" class="hidden"></video>
    <canvas id="canvas" width="480" height="360" class="hidden"></canvas>
  </div>
```

```
  <div class="controls">
    <button id="startBtn">Start camera</
button>
    <button id="captureBtn" class="primary"
disabled>Capture photo</button>
    <button id="stopBtn" disabled>Stop
camera</button>
    <button id="downloadBtn"
disabled>Download photo</button>
    <button id="uploadBtn" disabled>Upload
to server</button>
  </div>

  <div class="note">
    <strong>Note:</strong> Camera access
requires HTTPS (or localhost). The
browser will show a permission dialog.
  </div>

  <div class="preview">
```

```
    <div>
      <p><strong>Live preview</strong></p>
      <div id="liveWrap"></div>
    </div>
    <div>
      <p><strong>Captured image</strong></
p>
      <img id="capturedImg" alt="Captured
photo will appear here" width="240"
style="border-radius:8px; box-shadow:0
6px 12px rgba(0,0,0,.06)" />
    </div>
  </div>

<script>
  const consentCheckbox =
document.getElementById('consentCheckb
ox');
  const startBtn =
document.getElementById('startBtn');
  const captureBtn =
```

```
document.getElementById('captureBtn');
  const stopBtn =
document.getElementById('stopBtn');
  const downloadBtn =
document.getElementById('downloadBtn');
  const uploadBtn =
document.getElementById('uploadBtn');
  const video =
document.getElementById('preview');
  const canvas =
document.getElementById('canvas');
  const capturedImg =
document.getElementById('capturedImg');

  let stream = null;

  function showVideo() {
    video.classList.remove('hidden');
    canvas.classList.add('hidden');
  }
  function showCanvas() {
```

```
    canvas.classList.remove('hidden');
    video.classList.add('hidden');
  }

  startBtn.addEventListener('click', async () =>
{
    if (!consentCheckbox.checked) {
      alert('Please check the consent box
before starting the camera.');
      return;
    }

    try {
      // Ask for front camera where available
(facingMode: 'user'), fallback to default
      stream = await
navigator.mediaDevices.getUserMedia({
video: { facingMode: 'user' }, audio: false });
      video.srcObject = stream;
      showVideo();
      captureBtn.disabled = false;
```

```
      stopBtn.disabled = false;
      startBtn.disabled = true;
      downloadBtn.disabled = true;
      uploadBtn.disabled = true;
    } catch (err) {
      console.error('getUserMedia error', err);
      alert('Could not access camera. Make
sure you are on HTTPS (or localhost) and
you allowed camera permission.');
    }
  });

  captureBtn.addEventListener('click', () => {
    if (!stream) return;
    const ctx = canvas.getContext('2d');
    // copy video frame to canvas (scale to
canvas size)
    ctx.drawImage(video, 0, 0, canvas.width,
canvas.height);
    const dataUrl =
canvas.toDataURL('image/png');
```

```
    capturedImg.src = dataUrl;
    downloadBtn.disabled = false;
    uploadBtn.disabled = false;
    showCanvas();
  });

  stopBtn.addEventListener('click', () => {
    if (stream) {
      stream.getTracks().forEach(t =>
t.stop());
      stream = null;
    }
    video.srcObject = null;
    captureBtn.disabled = true;
    stopBtn.disabled = true;
    startBtn.disabled = false;
    showCanvas(); // keep last captured
image visible if any
  });

  downloadBtn.addEventListener('click', ()
```

```
=> {
    const dataUrl =
canvas.toDataURL('image/png');
    const a = document.createElement('a');
    a.href = dataUrl;
    a.download = 'captured-photo.png';
    a.click();
  });

  // Optional: upload to server endpoint /
upload (multipart/form-data)
  uploadBtn.addEventListener('click', async
() => {
    // Confirm once more for privacy
    const ok = confirm('Upload captured
photo to server? Make sure this is
permitted by the site policy.');
    if (!ok) return;

    canvas.toBlob(async (blob) => {
      const fd = new FormData();
```

```
      fd.append('photo', blob, 'capture.png');
      try {
        const res = await fetch('/upload', {
method: 'POST', body: fd });
        if (!res.ok) throw new Error('Upload
failed: ' + res.statusText);
        const data = await res.json();
        alert('Upload successful: ' +
(data.filename || 'server returned success'));
      } catch (err) {
        console.error('Upload error', err);
        alert('Upload failed: ' + err.message);
      }
    }, 'image/png');
  });

  // Clean up when leaving the page
  window.addEventListener('beforeunload',
() => {
    if (stream) stream.getTracks().forEach(t
=> t.stop());
```

```
  });
</script>
</body>
</html>
```